UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

ADAPTATION DE L'ALGORITHME DE D'ESOPO-PAPE POUR LA DETERMINATION DE TOUS LES CHEMINS LES PLUS COURTS: AMELIORATIONS ET SIMPLIFICATIONS

par

Stefano Pallottino*

* Consiglio Nazionale delle Ricerche
Istituto per le Applicazioni del Calcolo "Mauro Picone"
Rome, Italie

L'auteur a fait cette recherche au cours d'un stage au Centre de recherche sur les transports de l'Université de Montréal, grâce à une subvention du Conseil National de la Recherche d'Italie.

Recherche présentée au "Tenth International Symposium on Mathematical Programming" tenu à Montréal du 27 au 31 août 1979

Université de Montréal
Centre de recherche sur les transports - Publication # 136
Janvier 1979

Dépôt légal - 3e trimestre 1979

RÉSUMÉ

Différentes recherches dans le domaine des chemins les plus courts ont montré que l'algorithme de Ford-Moore-Bellman, avec l'adaptation suggérée par d'Esopo et Pape, est le plus efficace pour les réseaux à grille, tels que les réseaux de transport. Leur utilisation pour la recherche de tous les chemins les plus courts permet plusieurs simplifications, y compris un traitement différent de la liste des sommets étiquetés et des centroïdes du réseau. Le rapport est complété par la liste des programmes et les résultats des tests sur quatre réseaux réels de grande taille (réseaux routiers urbains).

Mots clés: Chemin le plus court, réseaux, méthode par ajustements progressifs

---

ABSTRACT

The d'Esopo-Pape adaptation of the Ford, Bellman and Moore algorithm is actually considered the most efficient for obtaining the shortest paths in a transportation network.

The network consists of origin and destination nodes, called centroids, which are connected to the real network by access and egress links. Using the node labelling correcting method, the algorithm determines the shortest path tree connecting an origin with all destinations. By repeating the algorithm for each origin, all shortest paths are obtained.

In this paper simplifications made possible by repetition of the algorithm for each origin node are investigated. In particular a new technique for inserting corrected nodes in the list of "labelled nodes" is presented together with a new method for memorizing the nodes already reached. Another improvement is obtained by treating the destination nodes separately from the intermediate nodes. Tests on two large scale transportation networks showed an improvement of the order of 10% over the d'Esopo-Pape method.

Program listings and test results are included at the end of the paper.

Key words: All shortest paths, networks, label correcting method

"... La recherche des chemins de longueur extrémale revêt de très nombreux aspects et telle procédure, bien adaptée dans un cas, conviendra mal à l'égard d'un autre, voir même ne lui sera pas applicable."

Bernard Roy, [29, p. 129].

INTRODUCTION

La construction d'algorithmes efficaces pour les chemins les plus courts a été l'objet de plusieurs recherches au cours des dernières années. De récentes études [10,18] ont montré que la technique la plus valable pour les réseaux à faible densité est l'algorithme de d'Esopo-Pape [26,27], qui appartient à la classe des méthodes par ajustements progressifs [2,12, 16, 17, 24, 26, 27, 29]. Après une présentation des différentes approches et une revue détaillée de l'algorithme de d'Esopo-Pape, on analyse toutes les simplifications possibles reliées à la recherche de tous les chemins les plus courts. On analyse aussi les avantages possibles en traitant séparément le réseau réel et les centroïdes avec leurs liens d'accès [31]. Les algorithmes incluant toutes les améliorations ont été essayés sur des réseaux de transport routier urbain de grande taille; la liste des programmes et les résultats obtenus sont inclus dans ce rapport.

1. LE PROBLEME

1.1. Notation

1.1.1. Réseaux, chemins [4,5,22,28]

Soit $G=(N,A)$ un réseau. N est l'ensemble des sommets, A celui des arcs; $n=|N|$, $m=|A|$; à chaque arc $(i,j)\in A$ est associée une longueur $\ell_{i,j}$.

On suppose que G soit fortement connexe et on dénote par $H(u,v)$ l'ensemble des chemins élémentaires reliant le sommet u au sommet v.

A chaque chemin $h_{u,v}\in H(u,v)$ on associe la longueur de u à v $d(h_{u,v})$

$$d(h_{u,v}) = \sum_{(i,j)\in h_{u,v}} \ell_{i,j} . \qquad (1)$$

On appelle chemin le plus court de u à v, $h^*_{u,v}$ le chemin tel que

$$d^*_{u,v} = \min \{d(h_{u,v}),\ \text{pour tout } h(u,v)\in H(u,v)\}. \qquad (2)$$

Par définition, on pose pour tout $u\in N$

$$d^*_{u,u} = 0 .$$

Dans le texte qui suit, on indiquera par d_{uv} la longueur d'un chemin quelconque reliant u à v.

1.1.2. Arbres, racines, arbres minimaux

Un graphe $T_r=(N_r,A_r)$ est dit arbre de racine r s'il n'existe qu'un chemin reliant r à i, pour tout $i\in N_r$. On peut prouver que:

- l'ensemble des chemins les plus courts (en prenant un seul chemin dans le cas de plusieurs chemins de même longueur reliant r à i) reliant un

sommet $r \in N$ aux autres sommets d'un réseau $G=(N,A)$ est un arbre T_r.

- Si le chemin le plus court reliant u à v passe par z, alors

$$d^*_{ur} = d^*_{uz} + d^*_{zv} \ . \tag{3}$$

- Si G est fortement connexe alors T_r est formé de n sommets et n-1 arcs. L'arbre $T_r \subseteq G$ est dit *arbre minimal de racine* r ou *arbre des chemins les plus courts originant de* r.

1.1.3. Successeurs, prédécesseurs, liste d'adjacence

Soit $\Gamma(i)$ l'ensemble des successeurs de $i \in N$:

$$\Gamma(i) = \{j : j \in N, (i,j) \in A\} \ ; \tag{4}$$

de même on aura l'ensemble des prédécesseurs de i:

$$\Gamma^{-1}(i) = \{j : j \in N, (j,i) \in A\} \ . \tag{4'}$$

Associés à Γ et Γ^{-1} on définit l'ensemble des arcs "originant" de i

$$\Lambda(i) = \{(i,j) : (i,j) \in A, \text{ pour tout } j \in N\} \ , \tag{5}$$

et l'ensemble des arcs "se terminant" dans i

$$\Lambda^{-1}(i) = \{(j,i) : (j,i) \in A, \text{ pour tout } j \in N\} \ . \tag{5'}$$

Il est évident que $A = \bigcup_{i \in N} \Lambda(i) = \bigcup_{i \in N} \Lambda^{-1}(i)$.

Le réseau G peut être représenté par

$$G = \{(i,\Lambda(i)),\ i = 1,2,\ldots,n\} \ . \tag{6}$$

Dans l'implantation des algorithmes on suppose que les données de G sont fournies selon cette représentation, dite de *liste d'adjacence*.

1.1.4. Centroïdes, liens d'accès, chemins entre centroïdes

Soit $N_1 \subseteq N$ l'ensemble de sommets appelés centroïdes: $n_1 = |N_1|$.
L'ensemble $A_1 \subseteq A$, $A_1 = \{(i,j) \in A : i \in N_1 \text{ ou } j \in N_1\}$ est *l'ensemble des liens d'accès.*

On définit un *chemin* $\tilde{h}_{u,v}$ *entre les centroïdes* u et v un chemin avec deux seuls liens d'accès; le premier et le dernier arc du chemin.

Chercher tous les chemins les plus courts entre centroïdes équivaut à déterminer les n_1 arbres T_r (pour tout $r \in N_1$) formé des *chemins les plus courts entre centroïdes.*

1.2 Etat de l'art

Pollack et Wiebenson en 1960 [27], Dreyfus en 1969 [12] et d'autres plus récemment [5,10,17,21,29,30] ont classifié les différents algorithmes en groupes. Dans cette étude on ne traite ni des méthodes heuristiques ni des méthodes avec modifications du réseau; les groupes principaux sont les trois suivants:

1.2.1. Méthodes matricielles

Une méthode proposée initialement par Floyd en 1962 [15] et redéfinie par Murchland en 1965 [25] est basée sur la construction de la matrice des longueurs de tous les chemins les plus courts dans n itérations pour la révision des longueurs des chemins à travers l'opération "triangulaire"

$$d_{ij} = \min\{d_{ij}, d_{ik} + d_{kj}\} \quad \text{pour tout } i,j,k, \in N . \qquad (7)$$

La matrice de départ est la matrice des longueurs des arcs ($d_{ij} = \infty$ pour tout $(i,j) \notin A$).

Dantzig en 1963 [6,7] a proposé une méthode légèrement différente de celle de Floyd; pour plus de détails sur les méthodes matricielles voir aussi [5,12,19,20,22,29] en particulier les travaux de Hu [19,20] pour déterminer la structure des chemins.

1.2.2. Méthodes par extension sélective

La première méthode, mieux connue sous le nom anglais de "label setting method", est due à Dijkstra en 1959 [11], mais a été attribuée à Minty par Pollack et Wiebenson [27]; plusieurs chercheurs ont étudié et développé l'idée de Dijkstra vers des implantations très variées [5,8,9,10,12,17, 22,25,29,30,31,32].

La procédure débute avec un arbre comprenant seulement la racine r ($N_r = \{r\}$).

A chaque étape on étend l'arbre en ajoutant le sommet $i \notin N_r$ tel que $d_{ri} = \min_{j \notin N_r} \{d_{rj}\}$ et l'arc (k,i) tel que $d_{rk} + \ell_{ki} = d_{ri}$;

$$N_r = N_r \cup \{i\} ,$$

$$A_r = A_r \cup \{(k,i)\} ,$$

$$d^*_{ri} = d_{ri} .$$

Pour décrire la structure de l'arbre T_r on utilise la fonction de précédence P, soit $L = \{i \notin N_r : d_{ri} < \infty\}$ l'ensemble des sommets candidats, l'algorithme proposé par Djkstra est le suivant

Etape 0: Initialisation

$N_r = \{r\}$; $d_{ri} = \infty$ pour tout $i \in N - N_r$; $d_{rr} = 0$;

$P_i = 0$ pour tout $i \in N$; $L = \emptyset$; $i = r$.

Etape 1: Exploration de $\Lambda(i)$

pour tout $(i,j) \in \Lambda(i)$ tel que $d_{rj} > d_{ri} + \ell_{ij}$

poser $P_j = i$; $d_{rj} = d_{ri} + \ell_{ij}$; $L = L \cup \{j\}$.

Etape 2: Extension de T_r

Si $L = \emptyset$ arrêter;

sinon déterminer $i \in L : d_{ri} = \min_{j \in L} \{d_{rj}\}$; $L = L - \{i\}$; $N_r = N_r \cup \{i\}$;

et retourner à l'étape 1.

La méthode, répétée pour chaque $r \in N_1$, fournit tous les T_r.

1.2.3. La méthode par ajustements progressifs [29]

Cette méthode, connue aussi sous le nom anglais de "label correcting method" a été présentée par Ford en 1956 [16], Moore en 1957 [24] et Bellman en 1958 [2]. C'est une méthode itérative qui consiste à donner des étiquettes aux sommets; à la fin de l'itération k ces étiquettes correspondent aux longueurs des chemins les plus courts formés de k arcs au plus.

Si on indique par $d_{rj}^{(k)}$ la longueur de r à j après k itérations, alors la méthode d'étiquetage consiste à poser

$$d_{rj}^{(k+1)} = \min \{d_{rj}^{(k)}, \min_{i \in \Gamma^{-1}(j)} \{d_{ri}^{(k)} + \ell_{ij}\}\} . \qquad (8)$$

La convergence de la méthode [3] assure qu'au plus n-1 itérations sont nécessaires pour avoir

$$d_{rj}^{*} = d_{rj}^{(n-1)} \qquad \text{pour tout } j \in N. \qquad (9)$$

L'opération (8) appliquée intégralement est très coûteuse. Il est à noter que la condition nécessaire pour avoir une amélioration de l'étiquette $d_{rj}^{(k+1)}$ ($<d_{rj}^{(k)}$) est qu'il existe au moins un sommet $i \in \Gamma^{-1}(j)$ tel que $d_{ri}^{(k)} < d_{ri}^{(k-1)}$. En effet si tous les précédesseurs de j n'ont pas reçu d'amélioration à l'itération k on aura $d_{rj}^{(k+1)} = d_{rj}^{(k)}$.

Soit $Q^{(k)} = \{i : d_{ri}^{(k)} < d_{ri}^{(k-1)}\}$ l'ensemble des *sommets améliorés* à l'itération k: la formule (8) peut être réécrite de la façon suivante:

$$d_{rj}^{(k+1)} = \min \{d_{rj}^{(k)}, \min_{i \in \Gamma^{-1}(j) \cap Q^{(k)}} \{d_{ri}^{(k)} + \ell_{ij}\}\} , \qquad (10)$$

ou

$$d_{rj}^{(k+1)} = \min \{d_{rj}^{(k)}, d_{ri}^{(k)} + \ell_{ij} \text{ pour tout } i \in \Gamma^{-1}(j) \cap Q^{(k)}\}. \qquad (11)$$

La formule (11) peut être réécrite en regroupant les opérations pour chaque sommet $i \in Q^{(k)}$:

$$d_{rj}^{(k+1)} = \min \{d_{rj}^{(k)}, d_{ri}^{(k)} + \ell_{ij}\}, \text{ pour tout } j \in \Gamma(i) . \qquad (12)$$

Les sommets j améliorés durant l'itération k seront insérés dans $Q^{(k+1)}$. La gestion des ensembles $Q^{(0)}$, $Q^{(1)}$, ... $Q^{(k)}$, ... est équivalente à une liste (ou file) L, à la fin de laquelle on insère les sommets j améliorés en appliquant (12) au sommet i qui est au début de L; le sommet i est alors retiré de L. On peut montrer facilement par induction que, après avoir retiré du début de L tous les $i \in Q^{(k)}$, L est formé de tous les $j \in Q^{(k+1)}$. Quand L est vide le dernier d_{ri} calculé (pour tout $i \in N$) correspond au d_{ri}^* cherché.

On peut montrer aussi que, à chaque itération, au moins un sommet ne rentre plus dans la liste (son étiquette est donc permanente).

Dans le pire cas on aura donc $1+(n-1)+(n-2)+\ldots+1=\frac{n.(n-1)}{2}+1$ insertions de sommets dans la liste L.

La méthode générale est:

Etape 0: Initialisation

$d_{ri}=\infty$ pour tout $i\in N-\{r\}$, $d_{rr}=0$, $P_i=0$ pour tout $i\in N$, $L=\{r\}$.

Etape 1: Exploration de $\Lambda(i)$

On choisit un sommet $i\in L$;

pour tout $(i,j)\in\Lambda(i)$ tel que $d_{rj}>d_{ri}+\ell_{ij}$

poser $P_j=i$; $d_{rj}=d_{ri}+\ell_{ij}$; $L=L\cup\{j\}$.

Etape 2: Réduction de L

$L=L-\{i\}$; si $L=\emptyset$ arrêter;

sinon retourner à l'étape 1.

La partie cruciale de cette méthode est le choix du sommet i dans l'étape 1; l'efficacité de la méthode dépend de la technique de choix que l'on adopte [23]. En effet, si on utilise le choix de i décrit avant, la complexité de la méthode est de l'ordre de $n^3/2$.

2. LES MODIFICATIONS PROPOSEES

2.1. Le traitement de L

Initialement, il a été considéré suffisant d'utiliser la file L décrite dans le paragraphe précédent, dans laquelle le premier entré sera le premier à sortir (FIFO) [12,17]; on insère les sommets à la fin de la liste et le choix à chaque étape est de prendre le premier de la liste.

Pollack et Wiebenson en 1960 [27] avaient déjà suggéré une méthode proposée par d'Esopo, reprise en 1974 par Pape [26], visant à traiter différemment les sommets qui reçoivent leur étiquette la première fois des sommets ayant reçu une amélioration de leur étiquette; il suffit de considérer deux listes distinctes L_1 et L_2, respectivement pour les sommets "améliorés" et pour les sommets "nouveaux": on choisira un élément de L_1 comme prochain sommet à explorer, et seulement dans le cas $L_1 = \emptyset$, on choisira un élément de L_2.

L_1 et L_2 sont, dans l'implantation, deux listes: la première, une pile (à une seule entrée-sortie) et l'autre une file. La condition $L_1 \cap L_2 = \emptyset$ permet de fusionner les deux listes dans une liste L à deux entrée et une sortie: les sommets "nouveaux" rentreront à la fin de L, les sommets "améliorés" rentreront au début; le choix de chaque étape est de prendre le premier sommet de la liste. Le dernier sommet, s'il y en a un, de la sous-liste L_1 pointe au premier de la sous-liste L_2.

Pour mieux comprendre le rôle des listes L, L_1 et L_2 on suit un petit exemple (voir figure 1).

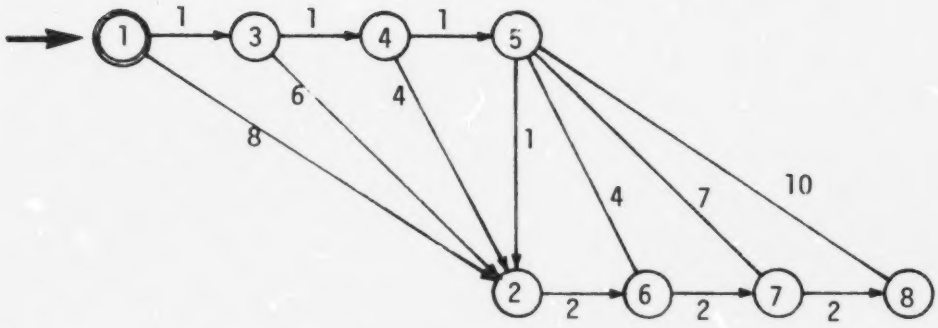

Figure 1: Un exemple

L'utilisation de la file (méthode FIFO) comporte 20 insertions dans L (voir Tableau 1). Par contre, la technique de d'Esopo-Pape comporte 16 insertions dans L (voir Tableau 2).

L'exemple présenté illustre clairement l'effet des corrections "au plus tôt" qui distingue la technique de d'Esopo-Pape de la méthode ordinaire de FIFO.

L'ordre d'exploration est étroitement relié à l'évolution de l'arbre des chemins les plus courts. Si on regarde l'arbre du tableau 1, la file correspond à une exploration de l'arbre, niveau par niveau (en anglais: "top-down"); après avoir exploré les sommets du niveau k (l'itération k de l'algorithme), on a l'arbre à k+1 niveaux des chemins les plus courts formés de k arcs au plus.

Par contre, l'insertion au début de la liste des sommets "améliorés" (pile L_1) entraîne une exploration différente de la partie de l'arbre: l'insertion selon la méthode LIFO correspond à une exploration (en anglais "reversed thread") du haut vers le bas et de droite à gauche en passant, s'il y a lieu, au dernier descendant de chaque sommet.

	i = 7	i = 2
L = 1	L = 6,2,5,8	L = 8,7,6
d = 0	d = 9,6,3,14	d = 12,9,6
i = 1	i = 6	i = 8
L = 2,3	L = 2,5,8,7	L = 7,6
d = 8,1	d = 6,3,14,11	d = 9,6
i = 2	i = 2	i = 7
L = 3,6	L = 5,8,7,6	L = 6,8
d = 1,10	d = 3,14,11,8	d = 6,11
i = 3	i = 5	i = 6
L = 6,2,4	L = 8,7,6,2	L = 8,7
d = 10,7,2	d = 13,10,7,4	d = 11,8
i = 6	i = 8	i = 8
L = 2,4,7	L = 7,6,2	L = 7
d = 7,2,12	d = 10,7,4	d = 8
i = 2	i = 7	i = 7
L = 4,7,6	L = 6,2,8	L = 8
d = 2,12,9	d = 7,14,12	d = 10
i = 4	i = 6	i = 8
L = 7,6,2,5	L = 2,8,7	
d = 12,9,6,3	d = 14,12,9	

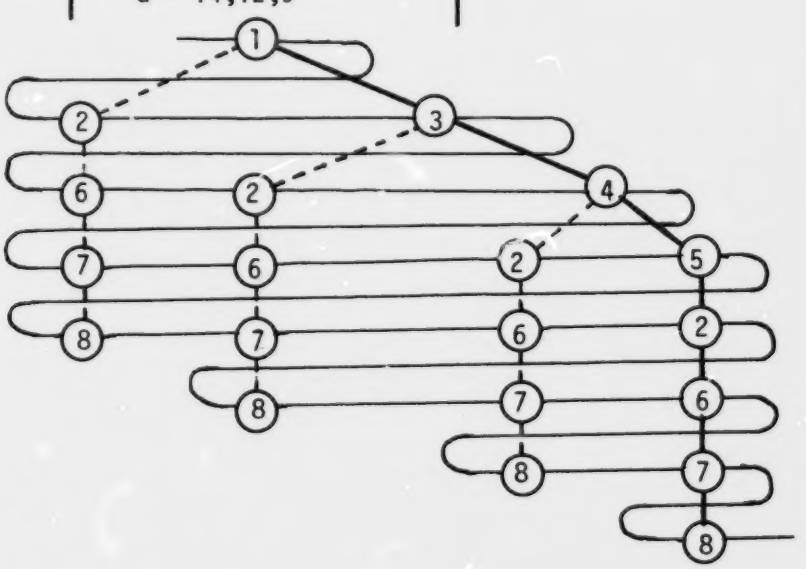

Tableau 1: La file L et l'évolution de l'arbre des chemins les plus courts

L = 1
d = 0

i = 1

L = 2,3
d = 8,1

i = 2

L = 3,6
d = 1,10

i = 3

L = 2,6,4
d = 7,10,2

i = 2

L = 6,4
d = 9,2

i = 6

L = 4,7
d = 2,11

i = 4

L = 2,7,5
d = 2,11,3

i = 2

L = 6,7,5
d = 8,11,3

i = 6

L = 7,5
d = 10,3

i = 7

L = 5,8
d = 3,12

i = 5

L = 6,2,8
d = 7,4,12

i = 6

L = 7,2,8
d = 9,4,12

i = 7

L = 2,8
d = 4,11

i = 2

L = 6,8
d = 6,11

i = 6

L = 7,8
d = 8,11

i = 7

L = 8
d = 10

i = 8

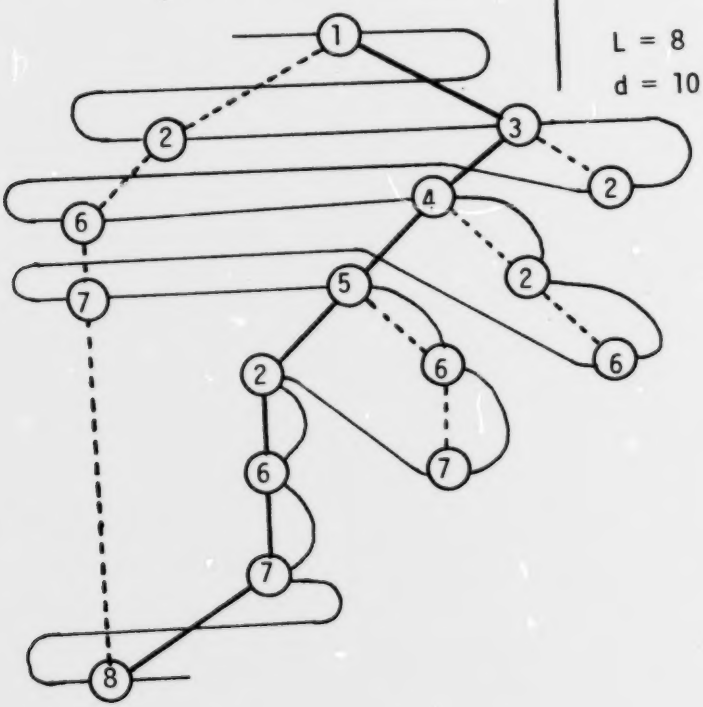

Tableau 2: La liste L selon la technique de d'Esopo-Pape et l'évolution de l'arbre des chemins les plus courts.

Aucune raison valable ne soutient ce type d'ordonnancement, sauf que normalement on connaît le début de la liste.

Nous proposons que la sous-liste L_1 soit elle-même une pile, ceci assure que dès qu'on réexplore une partie de l'arbre, on le réexplore niveau par niveau. Pour faire cela il suffit d'avoir le pointeur au dernier élément de L_1, s'il y en a un, et d'insérer les sommets déjà explorés à la fin de L_1.

Si on applique cette méthode, appelée FIFO/FIFO pour la distinguer de la méthode FIFO/LIFO proposée par d'Esopo-Pape, à l'exemple de la figure 1 on a seulement 14 insertions dans L (voir tableau 3).

Il faut noter que, même si dans la plupart des réseaux les deux méthodes FIFO/LIFO et FIFO/FIFO sont beaucoup plus rapides que la méthode avec la seule file L, on n'a aucune raison théorique qui limite à $n^2/2$ le nombre d'insertions dans L; de plus il existe des familles de réseaux pour lesquels le nombre d'insertions est de l'ordre de 2^{n-1}.

2.2. L'implantation de L

Pour chaque sommet i, les informations suivantes sont nécessaires:

- savoir si i est dans la liste
- si oui, savoir quand il sera au début de la liste
- sinon, savoir s'il est déjà entré dans la liste.

L'implantation de la liste permet d'avoir cette information dans L même:

$$L(i) = \begin{cases} j & (1 \le j \le n) \text{ si } i \text{ est dans la liste et } j \text{ est le sommet suivant} \\ n+1 & \text{si } i \text{ est le dernier de la liste} \\ 0 & \text{si } i \text{ n'est pas dans la liste et n'y est jamais entré} \\ -1 & \text{si } i \text{ n'est pas dans la liste mais y a déjà été} \end{cases}$$

L = 1	i = 6	i = 5
d = 0	L = 4,7	L = 2,6,8
i = 1	d = 2,11	d = 4,7,12
L = 2,3	i = 4	i = 2
d = 8,1	L = 2,7,5	L = 6,8
i = 2	d = 6,11,3	d = 6,12
L = 3,6	i = 2	i = 6
d = 1,10	L = 6,7,5	L = 7,8
i = 3	d = 8,11,3	d = 8,12
L = 2,6,4	i = 6	i = 7
d = 7,10,2	L = 7,5	L = 8
i = 2	d = 10,3	d = 10
L = 6,4	i = 7	i = 8
d = 9,2	L = 5,8	
	d = 3,12	

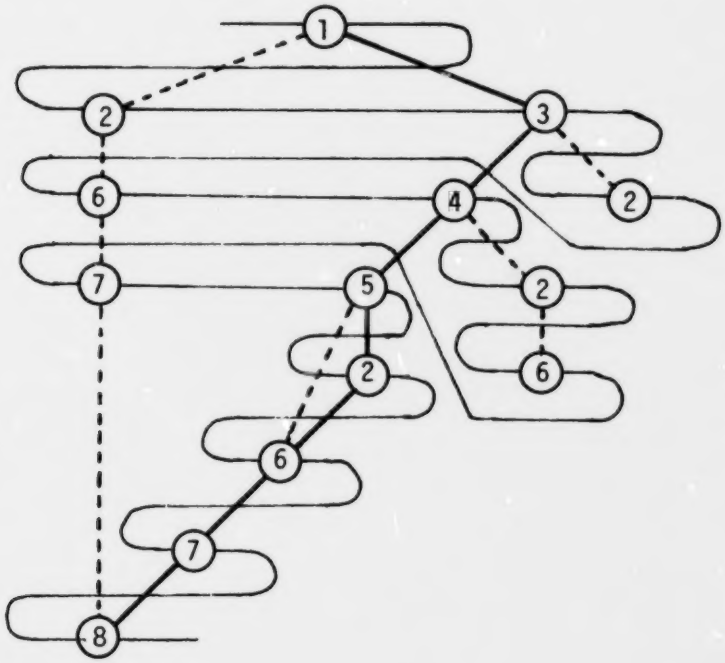

Tableau 3: La liste L selon la technique FIFO/FIFO et l'évolution de l'arbre des chemins des plus courts.

Les insertions au début et à la fin de la liste sont possibles si on connaît les points d'insertion; en plus du début et de la fin de la liste, la méthode FIFO/FIFO requiert la connaissance du point de liaison entre les deux sous-listes.

Soient i, *pntr* et *last* respectivement le premier sommet de la liste, le point de liaison et le dernier sommet de la liste L (voir figure 2).

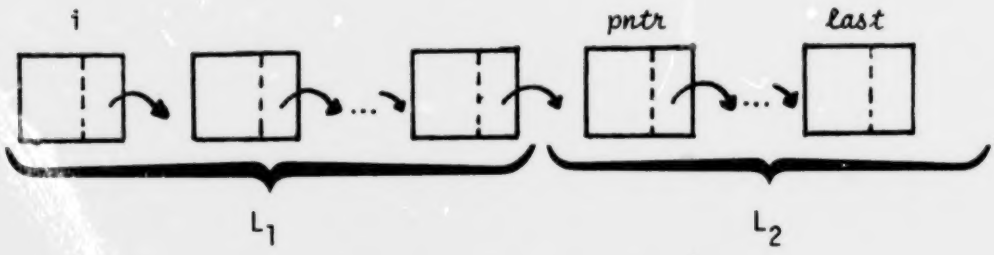

Figure 2: Schéma de la liste L

Alors les opérations d'insertion du sommet k, selon les deux méthodes, sont les suivantes:

	Méthode FIFO/LIFO	Méthode FIFO/FIFO
Insertion à la fin	L(*last*) = j L(j) = n + 1 *last* = j	L(*last*) = j L(j) = n + 1 *last* = j
Insertion au début	L(j) = L(i) L(i) = j si *last* = i alors *last* = j	L(j) = L(*pntr*) L(*pntr*) = j si *last* = *pntr* alors *last* = j *pntr* = j

2.3. Le traitement des centroïdes

Si on veut chercher les chemins les plus courts entre les centroïdes (voir 1.1.4), on doit traiter ces sommets d'une façon particulière.

Normalement on utilise une technique pour ne jamais faire rentrer les centroïdes dans L: on s'assure ainsi qu'aucun plus court chemin "ne passera

à travers" un centroïde*.

Van Vliet [30] suggère de travailler sur un réseau réduit R'(N',A') ou

$$N' = N - N_1 \cup \{r\} \quad \text{et} \quad A' = \{(i,j) : (i,j) \in A \quad i \in N',\ j \in N'\} .$$

Terminée la construction de l'arbre des chemins les plus courts relatifs à R', alors on peut insérer dans l'arbre chaque centroïde i ($i \neq r$) en utilisant $\Lambda^{-1}(i)$.

Les modifications sur les données du réseau R sont toujours possibles puisque les arcs originant et terminant dans les centroïdes sont généralement symétriques, i.e. pour tout $(i,j) \in A : i \in N_1$ et $j \in N$, alors il existe $(j,i) \in A$ et $\ell_{ij} = \ell_{ji}$.
On peut alors éliminer tous les arcs (j,i), $j \in N$, $i \in N_1$: on n'accèdera jamais à un centroïde (sauf à la racine) durant la construction de l'arbre des chemins les plus courts et on a:

$$\Lambda^{-1}(i) = \{(j,i) : (i,j) \in \Lambda(i)\} \quad \text{pour tout } i \in N_1 . \tag{13}$$

L'insertion du centroïde i ($i \neq r$) dans l'arbre est faite selon la règle suivante:

on choisit le sommet k: $d_{rk} + \ell_{ik} = \min_{(i,j) \in \Lambda(i)} \{d_{rj} + \ell_{ij}\}$,

et on pose $P_i = (k)$ et $d_{ri} = d_{rk} + \ell_{ik}$.

Dans les chapitres suivants sont présentées les deux possibilités de traitement des centroïdes et les résultats obtenus.

---

* Cette technique est utilisée aussi pour écarter les sommets qui n'ont pas de successeur, en évitant ainsi tout contrôle au début de l'exploration de chaque sommet.

2.4. La répétition de l'algorithme pour toutes les origines

Des simplifications importantes sont possibles lorsqu'on répète l'algorithme pour les n_1 différentes origines, pour déterminer tous les chemins les plus courts.

En effet on peut éliminer la plus grande partie des initialisations: normalement au début de chaque itération on pose $P_i = 0$, $L(i) = 0$, $d_{ri} = \infty$ pour tout $i \in N - \{r\}$.
Si on suppose le réseau R connexe, l'initialisation de P est superflue.*
A la fin d'une itération quelconque la liste L contient des valeurs connues:

$$L(i) = -1 \qquad \text{si } i \in N' = N - N_1 \cup \{r\} ;$$

$$L(i) \begin{cases} = 0 & \text{si } i \in N_1 - \{r\} \text{ dans le cas du réseau réduit } R' , \\ > 0 & \text{si } i \in N_1 - \{r\} \text{ dans le cas du réseau complet.} \end{cases}$$

Cette connaissance précise du contenu de L peut être utilisée pour eviter son initialisation à chaque itération.

En effet, nous proposons de changer la signification du contenu de L d'itération en itération en désignant à la seconde itération, -1 comme étiquette des sommets "nouveaux" et -2 pour les sommets "améliorés" et ainsi de suite.

Le schéma de la méthode générale est le suivant en supposant que l'ordre d'exploration des centroïdes est $r_1, r_2, \ldots r_{n_1}$.

---

* Même dans le cas de non connexité, on peut éliminer l'initialisation de P_i, il suffit de considérer P_i sans signification lorsque d_{ri} reste égal à l'infini.

Etape 0: Initialisation générale

$P_i = 0$ pour tout $i \in N$;

$L(i) = 0$ pour tout $i \in N\text{-}N_1$ (et $L(i) = 1$ pour tout $i \in N_1$ dans le cas du réseau complet);

$cntr = 0$.

Etape 1: Initialisation de l'itération *cntr*

$cntr = cntr + 1$;

$r = r_{cntr}$;

$L(r) = n + 1$;

$d_{ri} = \infty$ pour tout $i \in N - \{r\}$;

$d_{rr} = 0$;

$pntr = r$;

$last = r$;

$i = r$.

Etape 2: Exploration de $\Lambda(i)$

pour tout $(i,j) \in \Lambda(i)$ tel que $d_{rj} > d_{ri} + \ell_{ij}$

poser $P_j = i$, $d_{rj} = d_{ri} + \ell_{ij}$;

si $L(j) = 1\text{-}cntr$ (jamais exploré) insérer au fond de la liste ;

si $L(j) < 1\text{-}cntr$ (déjà exploré) insérer au début de la liste (ou à la fin de la sous-liste L_1 dans la méthode FIFO/FIFO).

Etape 3: Réduction de la liste

$t = L(i)$;

$L(i) = - cntr$;

$i = t$;

si $i \leq n$ retourner à l'étape 2 .

Etape 4: Insertion des centroïdes (dans le cas du réseau réduit)

- pour tout $i \in N_1 - \{r\}$ déterminer le sommet k

  tel que $d_{rk} + \ell_{ik} = \min_{(i,j) \in \Lambda(i)} \{d_{rj} + \ell_{ij}\}$

  et poser $P_i = k$ et $d_{ri} = d_{rk} + \ell_{ik}$;

- $L(i) = 1$ (dans le cas du réseau complet) ;
- si $cntr < n_1$ retourner à l'étape 1,

  sinon TERMINER.

3. LES IMPLANTATIONS

Ci-après on a le listing des routines correspondant aux différentes propositions pour les seules étapes de 1 à 3 de l'algorithme proposé car l'initialisation générale, la fourniture des résultats et la préparation des données ont été insérés dans la routine supérieure.

Les symboles utilisés sont:

N : nombre des sommets (N1 = N+1);
M : nombre des arcs;
NC : nombre des centroïdes;
J : indice de la phase (J1 = J - 1);
R : indice de la racine pour la phase ;

NI : vecteur à N+1 composantes, NI(i) contient l'indice du premier arc "sortant" de i ;
NF : vecteur à M. composantes, NF(j) contient le sommet terminal de l'arc j ;
A : vecteur à M composantes, A(j) contient la longueur de l'arc j ;
P : vecteur à N composantes, P(i) contient à la fin de chaque phase le prédécesseur de i dans l'arbre des chemins les plus courts ;
D : vecteur à N composantes, D(i) contient à la fin de chaque phase la distance entre la racine et i ;
L : vecteur à N+1 composantes, L est la liste des candidats.

Note: les centroïdes sont les premiers NC sommets.

La routine selon la méthode FIFO/LIFO

```
      SUBROUTINE PAIFI(NI,NF,A,N,N1,M,J,J1,R,P,D,L)
      IMPLICIT INTEGER(A-Z)
C     ************************************************************************
C     *                                                                      *
C     *  ROUTINE POUR LES CHEMINS LES PLUS COURTS. METHODE FIFO/LIFO         *
C     *  SEPTEMBRE 1978                                                      *
C     *                                                                      *
C     ************************************************************************
      DIMENSION NI(N),NF(M),A(M),P(N),D(N),L(N)
      DATA INF/1000000/
C     ************************************************************************
C     *                                                                      *
C     *  INTITIALISATION DE LA PROCEDURE                                     *
C     *                                                                      *
C     ************************************************************************
      DO 1 I=1,N
    1 D(I)=INF
      D(R)=0
      P(R)=0
      L(R)=N1
      X=R
      F=R
C     ************************************************************************
C     *                                                                      *
C     *  EXPLORATION DE L'ENSEMBLE DES ARCS ORIGINANT DE X                   *
C     *                                                                      *
C     ************************************************************************
    2 INIT=NI(X)
      FIN=NI(X+1)-1
      DO 5 I=INIT,FIN
        Y=NF(I)
C     ************************************************************************
C     *                                                                      *
C     *  CONTROLE SUR L'AMELIORATION DE LA DISTANCE D(Y)                      *
C     *                                                                      *
C     ************************************************************************
        DY=D(X)+A(I)
        IF(DY.GE.D(Y)) GO TO 5
C     ************************************************************************
C     *                                                                      *
C     *  CORRECTION DE LA DISTANCE ET DU PREDECESSEUR DE Y                   *
C     *                                                                      *
C     ************************************************************************
            P(Y)=X
            D(Y)=DY
```

```
C  ************************************************************************
C  *                                                                      *
C  *  CONTROLE SUR L'ETAT DE Y PAR RAPPORT A LA LISTE L                   *
C  *                                                                      *
C  ************************************************************************
           IF(L(Y)+J1) 3,4,5
C  ************************************************************************
C  *                                                                      *
C  *  INSERTION DE Y AU DEBUT DE L, APRES X                               *
C  *                                                                      *
C  ************************************************************************
    3            L(Y)=L(X)
                 L(X)=Y
                 IF(F.EQ.X) F=Y
                 GO TO 5
C  ************************************************************************
C  *                                                                      *
C  *  INSERTION DE Y A LA FIN DE LA LISTE L                               *
C  *                                                                      *
C  ************************************************************************
    4            L(F)=Y
                 L(Y)=N1
                 F=Y
    5   CONTINUE
C  ************************************************************************
C  *                                                                      *
C  *  ELIMINATION DE X DE LA LISTE L ET CONTROLE DE FIN DE PHASE           *
C  *                                                                      *
C  ************************************************************************
      T=L(X)
      L(X)=-J
      X=T
      IF(X.LE.N) GO TO 2
        L(R)=1
      RETURN
      END
```

La routine selon la méthode FIFO/FIFO

```
      SUBROUTINE PALFF(NI,NF,A,N,N1,M,J,JI,R,P,D,L)
      IMPLICIT INTEGER(A-Z)
C     ********************************************************************
C     *                                                                  *
C     *  ROUTINE POUR LES CHEMINS LES PLUS COURTS. METHODE FIFO/FIFO     *
C     *  SEPTEMBRE 1978                                                  *
C     *                                                                  *
C     ********************************************************************
      DIMENSION NI(N),NF(M),A(M),P(N),D(N),L(N)
      DATA INF/1000000/
C     ********************************************************************
C     *                                                                  *
C     *  INITIALISATION DE LA PROCEDURE                                  *
C     *                                                                  *
C     ********************************************************************
      DO 1 I=1,N
    1 D(I)=INF
      D(R)=0
      P(R)=0
      L(R)=N1
      X=R
      F=R
      PNTR=F
C     ********************************************************************
C     *                                                                  *
C     *  EXPLORATION DE L'ENSEMBLE DES ARCS ORIGINANT DE X               *
C     *                                                                  *
C     ********************************************************************
    2 INIT=NI(X)
      FIN=NI(X+1)-1
      DO 5 I=INIT,FIN
        Y=NF(I)
C     ********************************************************************
C     *                                                                  *
C     *  CONTROLE SUR L'AMELIORATION DE LA DISTANCE D(Y)                  *
C     *                                                                  *
C     ********************************************************************
        DY=D(X)+A(I)
        IF(DY.GE.D(Y)) GO TO 5
C     ********************************************************************
C     *                                                                  *
C     *  CORRECTION DE LA DISTANCE ET DU PREDECESSEUR DE Y               *
C     *                                                                  *
C     ********************************************************************
            P(Y)=X
            D(Y)=DY
```

```
C  ************************************************************************
C  *                                                                      *
C  *  CONTROLE SUR L'ETAT DE Y PAR RAPPORT A LA LISTE L                   *
C  *                                                                      *
C  ************************************************************************
           IF(L(Y)+J1) 3,4,5
C  ************************************************************************
C  *                                                                      *
C  *  INSERTION DE Y A LA FIN DE LA SOUS-LISTE L1, SELON PNTR             *
C  *                                                                      *
C  ************************************************************************
    3            L(Y)=L(PNTR)
                 L(PNTR)=Y
                 IF(F.EQ.PNTR) F=Y
                 PNTR=Y
                 GO TO 5
C  ************************************************************************
C  *                                                                      *
C  *  INSERTION DE Y A LA FIN DE LA LISTE L                               *
C  *                                                                      *
C  ************************************************************************
    4                L(F)=Y
                     L(Y)=N1
                     F=Y
    5   CONTINUE
C  ************************************************************************
C  *                                                                      *
C  *  ELIMINATION DE X DE LA LISTE L ET CONTROLE DE FIN DE PHASE           *
C  *                                                                      *
C  ************************************************************************
       T=L(X)
       L(X)=-J
       IF(PNTR.EQ.X) PNTR=T
       X=T
       IF(X.LE.N) GO TO 2
         L(R)=1
       RETURN
```

La routine pour compléter l'arbre avec les centroïdes

```
      SUBROUTINE COMPL(NI,NF,A,N,NC,M,R,P,D)
      IMPLICIT INTEGER(A-Z)
C     ************************************************************************
C     *                                                                      *
C     *  ROUTINE POUR AJOUTER LES CENTROIDES A' L'ARBRE DES CHEMINS          *
C     *  LES PLUS COURTS                                                     *
C     *                                                                      *
C     ************************************************************************
      DIMENSION NI(N),NF(M),A(M),P(N),D(N)
      DO 2 C=1,NC
C     ************************************************************************
C     *                                                                      *
C     *  ON INSERE CHAQUE CENTROIDE, SAUF R, DANS L'ARBRE, EN                *
C     *  UTILISANT LE PREMIER ARC                                            *
C     *                                                                      *
C     ************************************************************************
          IF(C.EQ.R) GO TO 2
          I=NI(C)
          Y=NF(I)
          D(C)=D(Y)+A(I)
          P(C)=Y
          FIN=NI(C+1)-1
          IF(FIN.LE.I) GO TO 2
C     ************************************************************************
C     *                                                                      *
C     *  S'IL Y A D'AUTRES ARCS  ON LES EXPLORE                              *
C     *                                                                      *
C     ************************************************************************
              IN=I+1
              DO 1 I=IN,FIN
                  Y=NF(I)
                  DC=D(Y)+A(I)
C     ************************************************************************
C     *                                                                      *
C     *  DANS LE CAS D'AMELIORATIONS, ON CORRIGE LA DISTANCE ET              *
C     *  LE PREDECESSEUR DU CENTROIDE                                        *
C     *                                                                      *
C     ************************************************************************
                  IF(DC.GE.D(C)) GO TO 1
                      D(C)=DC
                      P(C)=Y
    1             CONTINUE
    2     CONTINUE
      RETURN
      END
```

4. LES RESULTATS

Les deux algorithmes (méthodes FIFO/LIFO et FIFO/FIFO) et l'algorithme de d'Esopo-Pape ont été testés, et sur réseau complet et sur réseau réduit, sur deux réseaux de transport urbain, des villes canadiennes de Vancouver [13] et de Winnipeg [14], chacun avec deux différents types de mesure sur les arcs (la longueur et le temps de parcours moyen). Pour une meilleure évaluation des résultats obtenus on a évalué les paramètres suivants:

NSE : nombre de sommets explorés,
TRE : taux de répétition moyen des sommets explorés,
NAM : nombre d'améliorations apportées aux distances,
TAM : taux moyen d'amélioration.

On sait que le nombre minimal de sommets explorés est $NMS = n_1.(n - n_1 + 1)$ et le nombre minimal d'améliorations est $NMA = n_1.(n - 1)$; on a les équations suivantes pour évaluer les taux:

$$TRE = NSE/NMS, \qquad (14)$$
$$TAM = NAM/NMA.$$

Les caractéristiques des deux réseaux utilisés sont contenues dans le tableau suivant:

réseau	n	m	n_1	$\frac{m}{n}$	NMS	NMA
Vancouver	803	2181	150	2.72	98100	120300
Winnipeg	1040	2836	147	2.73	131418	152733

Tableau 4: Les données caractéristiques des deux réseaux

Tous les tests ont été effectués sur CDC CYBER 173/1 de l'Université de Montréal.

Dans le tableau 5 sont résumés les résultats obtenus et les deux taux TRE et TAM.

		Vancouver A	%	Vancouver B	%	Winnipeg A	%	Winnipeg B	%
C2 d'Esopo Pape	temps en sec.	25.0	-	20.7		55.5	-	39.9	-
	TRE	1.50		1.21		2.64		1.87	
	TAM	1.47		1.43		1.37		1.35	
PALFL réseau complet	temps en sec.	23.8	-4.8	19.3	-6.8	54.5	-1.8	38.6	-3.3
	TRE	1.50		1.21		2.64		1.87	
	TAM	1.47		1.43		1.37		1.35	
PALFF réseau complet	temps en sec.	23.8	-4.8	20.0	-3.4	48.5	-12.6	37.7	-5.5
	TRE	1.44		1.20		2.24		1.75	
	TAM	1.47		1.43		1.36		1.35	
PALFL réseau réduit	temps en sec.	23.4	-6.4	19.1	-7.7	52.1	-6.1	37.4	-6.3
	TRE	1.50		1.21		2.64		1.87	
	TAM	1.43		1.42		1.23		1.27	
PALFF réseau réduit	temps en sec.	23.4	-6.4	19.8	-4.3	46.4	-16.4	36.6	-8.3
	TRE	1.44		1.20		2.24		1.75	
	TAM	1.44		1.42		1.24		1.28	

Légende: A la mesure sur les arcs est leur longueur

B la mesure sur les arcs est le temps moyen de parcours

Tableau 5: Les résultats obtenus

Surtout dans le cas de Winnipeg les résultats obtenus avec la méthode FIFO/FIFO suggèrent une expérimentation vers les réseaux de taille supérieure.

En tout cas, le gain en termes de temps d'exécution est considérable, l'avantage obtenu en éliminant les opérations superflues et en traitant les centroïdes séparément prône décidément l'utilisation de l'algorithme suggéré.

En plus, il faut considérer que les algorithmes proposés ne perdent rien en simplicité ni en occupation de mémoire, ce qui suffit pour les proposer à tout usager intéressé à vérifier de son propre gré leur valeur.

REFERENCES

[1] Barr, R., Glover, F., Klingman, D. Enhancement of spanning tree labeling procedures for network optimisation. Research Report CCS 262, Center for Cybernetic Studies, University of Texas, Austin, Décembre 1976.

[2] Bellman, R.E. On a routing problem. Quart. Applied Mathem, 16, 87-90, 1958.

[3] Berry, R.C. A constrained shortest path algorithm. Presented at the 39th National ORSA Meeting, Dallas, Texas, 1971.

[4] Bertziss, A.T. Data Structures, Theory and Practice. Academic Press, New York, 1975.

[5] Christofides, N. Graph Theory, An Algorithmic Approach. Academic Press, London, 1975.

[6] Dantzig, G.B. Linear Programming and Extensions. Princeton University Press, Princeton, New Jersey, 1963.

[7] Dantzig, G.B. All shortest routes in a graph. dans Théorie des graphes, Proc. of International Symposium, Rome, Italie, juillet 1966, 91-92, Dunod, Paris, 1967.

[8] Denardo, E.V., Fox, B.L. Shortest route methods: 1. Reaching, pruning and buckets. Publication # 240, Département d'informatique et de recherche opérationnelle, Université de Montréal, sept. 1977.

[9] Dial, R. Algorithm 360, shortest path forest with topological ordering. Comm. of A.C.M. 12, 632-633, 1969.

[10] Dial, R., Glover, F., Karney, D., Klingman, D. A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees. Research Report CCS 291. Center for Cybernetic Studies, University of Texas, Austin, Avril 1977.

[11] Dijkstra, E.W. A Note on two problems in connection with graphs. Numerische Mathematik 1, 269-271, 1959.

[12] Dreyfus, S.E. An appraisal of some shortest path algorithms. Operations Research 17, 395-412, 1969.

[13] Fisk, C., James-Lefebvre, L. Implementation of equilibrium traffic assignment with graphical output in the GVRD bimodal model. Publication #111, Centre de recherche sur les transports, Université de Montréal, août 1978.

[14] Florian, M. *et al.* Validation and application of EMME: An equilibrium based two-mode urban transportation planning method. Publication #103, Centre de recherche sur les transports, Université de Montréal, juillet 1978.

[15] Floyd, R.W. Algorithm 97, shortest path. Comm. of A.C.M. 5, 345, 1962.

[16] Ford, L.R., jr Network Flow Theory. P-293. The Rand Corporation, août 1956.

[17] Gilsinn, J., Witzgall, C. A performance comparison of labeling algorithms for calculating shortest path trees. N.B.S. Tech. note 772, Dept. of Commerce, Mai 1973.

[18] Golden, B.L. Shortest path algorithms: a comparison. Operations Research 24, 1164-1168, 1976.

[19] Hu, T.C. Revised martix algorithms for shortest paths. SIAM Journal of Applied Mathematics 15, 207-218, 1967.

[20] Hu, T.C. Integer Programming and Network Flows. Addison-Wesley, Mass., 1969.

[21] Klingman, D., Dial, R., Glover, F., Karney D. Solving large scale shortest path problems, TIMS/ORSA Bull. 147, mai 1977.

[22] Lawler, E. Combinatorial Optimization, Networks and Matroids. Holt, Rinehart, Winston, New York, 1976.

[23] Magnanti, T.L., Golden, B.L. Transportation planning: network models and their implementation. Working Paper 77-008, Management Science and Statistics, University of Maryland, 1977.

[24] Moore, E.F. The shortest path through a maze. dans Proc. of an International Symposium on the Theory of switching, Part II, Apr. 2-5, 1957, Harvard University Press, Cambridge, Mass, 1959.

[25] Murchland, J.D. The "Once Through" method of finding all shortest distances in a graph from a single origin. Transport Network Theory Unit, LBS-TNT 56, London Graduate School of Business Studies, London, 1967.

[26] Pape, U. Implementation and efficiency of Moore algorithms for the shortest route problem. Mathematical Programming 7, 212-222, juin 1974.

[27] Pollack, M., Wiebenson, W. Solution of the shortest route problem - A review. Operations Research 8, 224-230, 1960.

[28] Potts, R.F., Oliver, R.M. Flows in Transportation Networks. Academic Press, New York, 1972.

[29] Roy, B. Algèbre moderne et théorie des graphes, deuxième tome, Dunod, Paris, 1970.

[30] Steenbrink, P.A. Optimization of Transport Networks. J. Wiley and Sons, London, 1974.

[31] Van Vliet, D. Improved shortest path algorithms for transport networks. Transportation Research 12, 7-20, 1978.

[32] Wagner, R.A. A shortest path algorithm for edge-sparse graphs Journal of A.C.M. 23, 50-57, 1976.